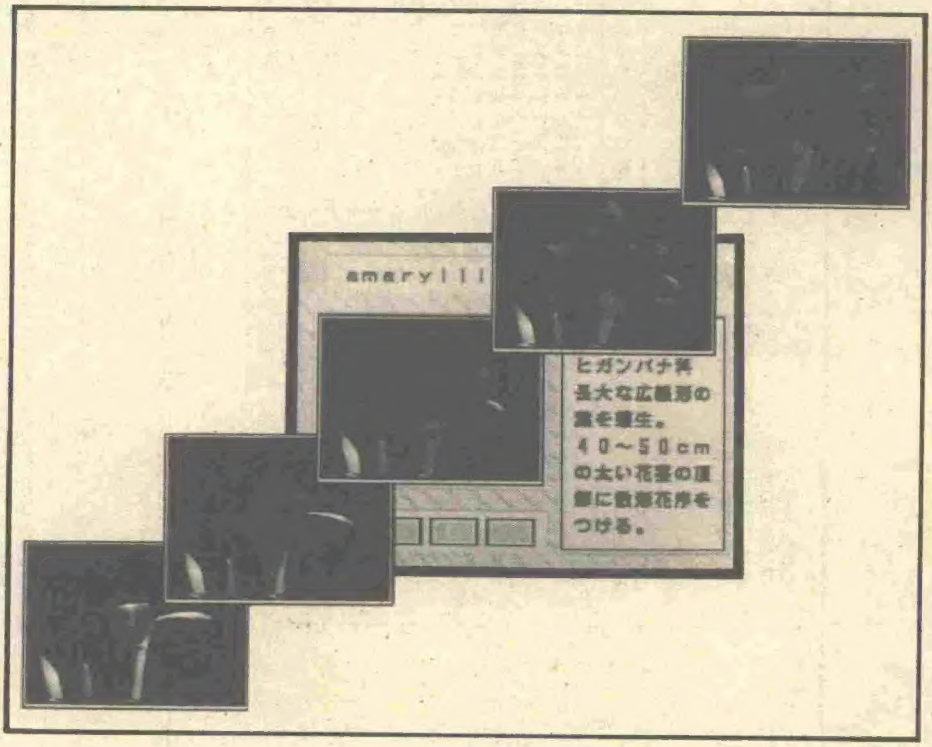
บุญเลิศ เอี่ยมทัศนา

TPW
: นิทรรศการทางจอภาพ

เทคนิคการเขียนโปรแกรมโดยใช้เทอร์โบปาสคาลบนวินโดวส์ เพื่อแสดงภาพในแบบการฉายสไลด์



ในงานนิทรรศการหรือแม้แต่ในการเรียนการสอน มักจะมีการแสดงภาพในแบบเครื่องฉายสไลด์ทางจอภาพ บางท่านอาจเคยสงสัยว่าเขาทำได้อย่างไร หากเขียนโปรแกรมบนดอสจะต้องใช้เวลามากพอสมควร ขอบคุณวินโดวส์ที่ทำให้เขียนโปรแกรมได้สั้นและเทอร์โบปาสคาลที่ทำให้เขียนได้ง่าย

เมื่อไม่เป็นรีซอร์ส

ในเรื่องไอคอนของขวัญ ผมได้แสดงการนำภาพบิตแม็พที่ทำเป็นรีซอร์สไว้ออกแสดง และเพราะทำเป็นรีซอร์สไว้วินโดวส์จึงได้ดำเนินการเกือบทั้งหมดแทนเรา ทำให้สามารถนำออกแสดงได้โดยง่าย แต่จะต้องเป็นบิตแม็พขนาดเล็ก (ขนาดของไฟล์เล็กกว่า 32 กิโลไบต์)

หากเป็นบิตแม็พขนาดใหญ่ จะทำเป็นรีซอร์สไม่ได้ (ทำได้แต่โหลดลงหน่วยความจำไม่ได้) เราจะต้องดำเนินการเอง ซึ่งมีปัญหาอยู่ 2 เรื่องคือ

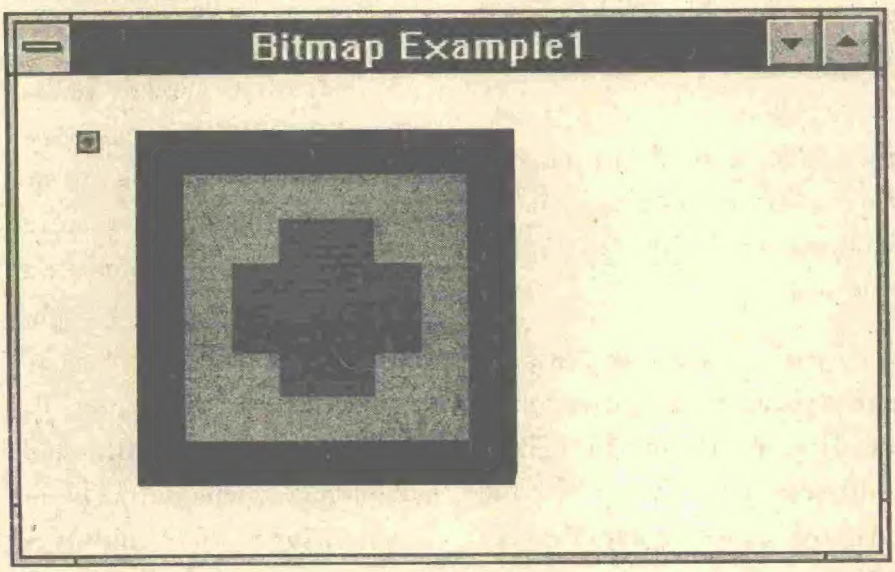
- การโหลดไฟล์ขนาดใหญ่
- การโหลดไฟล์ที่เป็นบิตแม็พ

โปรแกรมเสริมความเข้าใจ

โปรแกรมนี้จะยังไม่ตอบปัญหาดังกล่าวมา แต่จะแสดงว่าถ้าเราสร้าง

```
PROGRAM BitmapExample1;

USES  WinTypes,WinProcs,WObjects,Strings;

CONST appname  = 'bmpexp1';
      thetitle = 'Bitmap Example1';

TYPE
IOFunction = function(FP: integer; Buf: PChar; Size: Integer): Word;


TYPE
 WndProcCP = ^WndProcC;
 WndProcC = OBJECT(TWindow)

constructor  Init(apar : PWindowsObject; atitle : PChar);
procedure wmRButtonDown(var msg: TMessage);
          virtual wm_First + wm_RButtonDown;
          END;
```

ลิสติงที่ 1.1



รูปที่ 1 ผลการรันโปรแกรม BitmapExample1

ภาพบิตแม็พในโปรแกรมของเรา จะนำออกแสดงได้อย่างไร ดังโปรแกรมในลิสติง 1 ทั้งหลาย (1.1 ถึง 1.5) โดยมีส่วนการกำหนดสกุล WndProC ในลิสติง 1.1

เมื่อรันจะให้ผลการรันดังรูปที่ 1 โดยที่ในโปรแกรม ผมได้สร้างภาพขนาด 8×8 จุดภาพ ทำเป็นกรอบสีน้ำเงินตรงกลางเป็นกากบาทแดงบนพื้นสีเหลือง แต่ภาพเล็กมากผมจึงให้ภาพขยาย 16 เท่าอีกภาพหนึ่ง ดังที่เห็นในรูปที่ 1

โปรแกรมนี้ถึงจะสั้น แต่ก็มีเรื่องที่จะกล่าวถึงมากพอสมควร เพื่อเป็นพื้นฐานต่อไปครับ กล่าวคือ

ระบบ

ภาพที่จะวาดนี้เรียกว่า DIB (Device-Independent Bitmap) คือภาพที่ไม่ขึ้นกับอุปกรณ์ชนิดใดแบบใด เช่น สามารถนำไปแสดงผลได้ทั้งบนจอภาพและเครื่องพิมพ์ ซึ่งจอภาพอาจเป็นจอสีหรือโมโนโครม และเครื่องพิมพ์อาจเป็นดอตแมตริกซ์หรือเลเซอร์

โดยที่ในการนำออกแสดงจะต้องเขียนโปรแกรมทำภาพให้เป็น ซึ่งผมขอเรียกว่า แผ่นภาพ (device-specific memory bitmap) ให้เหมาะกับอุปกรณ์แต่ละชนิด เช่นสำหรับจอภาพ เป็นต้น ซึ่งจะใช้กับเครื่องพิมพ์ได้ด้วย เพราะถือว่าเป็น raster scan เหมือนกัน

แต่ไม่ต้องเขียนโปรแกรมสำหรับอุปกรณ์แต่ละแบบ เช่น สำหรับจอโมโนโครมหรือสำหรับจอสี เป็นต้น เพราะมีดีไวซ์ไดรเวอร์ของอุปกรณ์ต่าง ๆ เหล่านี้อยู่แล้ว ตามที่ผู้ใช้ติดตั้งกับเครื่องของตน

จานสี

เมื่อจะวาดภาพก็ต้องมีจานสี โดยที่จะต้องเลือกว่าจะใช้จานสีขนาดกี่ช่องสี ค่าโดยทั่วไปซึ่งจะเกี่ยวพันกับหน่วยความจำที่จะใช้เก็บสีของแต่ละจุดภาพด้วย คือ

- 2 สี 1 บิต
- 16 สี 4 บิต
- 256 สี 1 ไบต์

```
(* ------------------ Bitmap Data ------------------ *)

CONST
 numco = 16;

 maxco = 4;
 palette : array[0..maxco-1] of TRGBQuad =
  ((rgbBlue : 255; rgbGreen : 0; rgbRed : 0; rgbReserved : 0),
   (rgbBlue : 0; rgbGreen : 255; rgbRed : 0; rgbReserved : 0),
   (rgbBlue : 0; rgbGreen : 0; rgbRed : 255; rgbReserved : 0),
   (rgbBlue : 255; rgbGreen : 255; rgbRed : 255; rgbReserved : 0));


 wwid = 8;  hhi = 8;
 redcross : array[0..31] of Byte =
  ($00, $00, $00, $00,
   $01, $11, $11, $10,
   $01, $12, $21, $10,
   $01, $22, $22, $10,
   $01, $22, $22, $10,
   $01, $12, $21, $10,
   $01, $11, $11, $10,
   $00, $00, $00, $00);
```

ลิสติงที่ 1.2

- 2^{24} สี 3 ไบต์

ดังที่ในลิสติง 1.2 ได้กำหนดจานสีไว้ 16 ช่อง ตามค่าคงตัวที่ให้แก่ numco เพื่อใช้ในการเขียนโปรแกรมต่อไป

ผสมสี

เมื่อมีจานสีแล้วจะต้องผสมสีใส่ตามช่องต่าง ๆ เช่นเดียวกับช่างเขียนครับ เราไม่จำเป็นต้องใส่สีครบทุกช่อง (แต่ส่วนมากครบ) ดังโปรแกรมนี้ผมบรรจุสีไว้เพียง 4 ช่องเท่านั้น ดังค่าที่ให้แก่แพเลตต์ (หมายถึงจานสี)

สำหรับค่าที่ให้จะต้องให้เป็นสูตรผสมสีเป็นค่า RGB คือค่า แดง เขียว น้ำเงิน ซึ่งเมื่อผสมกันแล้วจะเป็นสีต่าง ๆ และเราจะให้ช่องใดเป็นสีใดก็ได้ ค่าที่ให้ได้ของแต่ละสีคือ 0 ถึง 255 ค่าที่ให้ไว้จะได้เป็นสี น้ำเงิน เขียว และแดง (ที่เป็นแม่สี) กับสีขาว (หากได้เป็น 0 ทั้งสามสีจะได้สีดำ) แต่ทั้งนี้ไม่ได้หมายความว่าเราจะได้สี 1 ใน 256×256×256 สี เพราะเรากำหนดหน่วยความจำไว้เพียง 4 บิตต่อ 1 จุดภาพ วินโดวส์จะนำไปทำสีใกล้เคียงให้เราได้สี 1 ใน 16 สีเท่านั้น

สร้างภาพ

นำไปสร้างภาพดังการให้ค่าคงตัว แก่ redcross ค่าที่ให้เป็นค่าอินเด็กซ์ของจานสีระบุว่าจะใช้สีช่องที่เท่าไรในจานสี ไบต์หนึ่งจึงเป็นตัวแทนของภาพได้ 2 จุด ภาพขนาด 8×8 จุดภาพ จึงใช้หน่วยความจำ 4×8 ไบต์

เช่นเดียวกับช่างเขียนอีกเช่นกัน คือ ถึงแม้จะบรรจุสีไว้ 4 ช่องก็ไม่จำเป็นต้องใช้หมดทั้ง 4 ช่อง ตามโปรแกรมนี้ผมใช้เพียง 3 สีเท่านั้นเพื่อแสดงแม่สี ค่าที่ให้แก่แต่ละจุดภาพจึงมีค่า 0 ถึง 2 เท่านั้น

ขนาดของภาพขึ้นกับเราครับว่าจะให้มีขนาดความกว้างเท่าไร และควรทำค่าคงตัวบ่งขนาดความกว้างและความสูงของภาพไว้ด้วย ดังค่าที่ให้แก่ wwid และ hhi

การสร้างภาพมีเพียงเท่านี้ หากยังไม่ต้องการนำออกแสดงสามารถเก็บลงไฟล์ไว้ได้ ซึ่งถ้าต้องการให้เป็นไฟล์ .BMP จะต้องเขียนโปรแกรมให้ ส่วนหัวของไฟล์ตามข้อกำหนดของวินโดวส์ตามที่จะกล่าวต่อไป

ทำเป็นแผ่นภาพ

เมื่อต้องการนำออกแสดงจะต้องทำเป็นแผ่นภาพก่อน ดังโปรแกรมของกิจกรรม wmRButtonDown (เพราะผมกำหนดไว้ว่าให้คลิ๊กปุ่มขวาของเม้าส์เมื่อต้องการให้แสดงภาพ) ในลิสติง 1.3

โปรแกรมส่วนนี้ผมนำมาจากคู่มือของไมโครซอฟต์ (Microsoft Windows 3.1 Programmer's Reference) ในการอธิบายคำว่า CreateDIBitmap โดยมีขั้นตอนในการทำแผ่นภาพดังนี้ คือ

1. เตรียมส่วนหัวของแผ่นภาพ คือ จองหน่วยความจำเท่าขนาดของ

```
(* ------------------ WndProC Method ------------------ *)

CONSTRUCTOR WndProC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
 TWindow.Init(apar,atitle);
 attr.x := 10; attr.y := 10; attr.w := 300; attr.h := 200;
END;


{ Create BMP }
PROCEDURE WndProC.wmRButtonDown(var msg: TMessage);
VAR  hInfo : THandle; adc : HDC;
    pInfo : PBitmapInfo; hBMP : HBitmap;
BEGIN
 hInfo := LocalAlloc(lmem_ZeroInit or lmem_Moveable,
       SizeOf(TBitmapInfoHeader) + (SizeOf(TRGBQuad) * numco));
 pInfo := PBitmapInfo(LocalLock(hInfo));
 with pInfo^.bmiHeader do
  begin biSize := SizeOf(TBitmapInfoHeader);
       biWidth := wwid; biHeight := hhi;
       biPlanes := 1; biBitCount := 4;
       biCompression := bi_RGB end;
 Move(palette, pInfo^.bmiColors, SizeOf(TRGBQuad) * maxco);
 adc := GetDC(0);
 hBMP := CreateDIBitmap(adc, pInfo^.bmiHeader, cbm_Init,
        @redcross, pInfo^, dib_RGB_Colors);
 ReleaseDC(0, adc); LocalFree(hInfo);
 DispBMPs(hWindow,hBMP,20,20,wwid,hhi,false,0,0,wwid,hhi);
 DispBMPs(hWindow,hBMP,40,20,wwid*16,hhi*16,true,0,0,wwid,hhi);
 DeleteObject(hBMP);
END;
```

ลิสติงที่ 1.3

เรคอร์ด TBitmapInfoHeader กับจานสีแล้วให้ค่าพอยน์เตอร์แก่ WInfo โดยที่คำในภาษาอังกฤษใกล้เคียงกันมากจึงขอเรียกเรคอร์ดนี้ว่า เรคอร์ดรอง

ส่วน pInfo เป็นพอยน์เตอร์ไปยังข้อมูลแบบ BitmapInfo ซึ่งจะขอเรียกว่า เรคอร์ดใหญ่ ทั้งนี้เพราะในเรคอร์ดใหญ่มีเรคอร์ดรองเป็นฟิลด์หนึ่งคือ ฟิลด์ bmiHeader และมีช่องแรกของจานสีเป็นอีกฟิลด์หนึ่ง คือ ฟิลด์ bmiColors ทั้งนี้เพื่อการอ้างแอดเดรสถึงเท่านั้น

2. ให้ค่าแก่เรคอร์ดรอง (ความจริงอยู่ในขั้นตอนเดียวกันกับข้อ 1 แต่มีเรื่องอื่นมาแทรกจึงขอแยก) ตามที่ให้แก่ pInfo .bmiHeader คือ

- biSize ขนาดของเรคอร์ดรอง (เพื่อการตรวจสอบ)
- biWidth และ biHeight เป็นค่าความกว้างและความสูงของภาพ
- biPlanes คือจำนวน plane ให้ค่าเป็น 1
- biBitCount จำนวนบิตต่อ 1 จุดภาพตามที่ทำเป็นภาพไว้
- biCompression ให้ค่าเป็น bi__RGB
- มีค่าอื่นอีกแต่ไม่จำเป็นต้องให้

3. ให้ค่าจานสีที่เตรียมไว้แก่ส่วนหัวของแผ่นภาพ (ทั้งนี้เพราะต้องการให้เป็นหน่วยความจำผืนเดียวติดต่อกัน)

4. ทำแผ่นภาพด้วย CreateDIBitmap ค่าที่ผ่านไปให้คือ

- ค่าดีซี ของอุปกรณ์ เช่น จอภาพ
- เรคอร์ดรอง
- ค่า cbm__Init (เป็นค่าคงตัว)
- แอดเดรสของภาพ
- เรคอร์ดใหญ่ เพื่อต้องการจานสี (แต่หาเหตุผลไม่ได้ว่า เมื่อให้เรคอร์ดใหญ่แล้วทำไมถึงต้องการเรคอร์ดรองด้วย)
- ค่า dib__RGB__Colors (ตามนี้)

```
PROCEDURE DispBMPs(hWndd : HWnd; hBMP : HBitmap;
                   dx,dy,dwid,dhi : Integer; stretch : Boolean;
                   sx,sy,swid,shi : Integer);
VAR hOldObj : THandle; wndDC,memDC : HDC;
BEGIN
 wndDC := GetDC(hWndd);
 memDC := CreateCompatibleDC(wndDC);
 hOldObj := SelectObject(memDC, hBMP);
 if dwid = 0 then dwid := swid;
 if dhi = 0 then dhi := shi;
 if stretch then
  StretchBlt(wndDC, dx,dy,dwid,dhi, memDC,sx,sy,swid,shi, srcCopy)
 else
  BitBlt(wndDC, dx,dy,dwid,dhi, memDC,sx,sy, srcCopy);
 ReleaseDC(hWndd, wndDC);
 SelectObject(memDC, hOldObj);
 DeleteDC(memDC);
END;
```

ลิสติงที่ 1.4

CreateDIBitmap จะทำแผ่นภาพให้เรา ซึ่งเป็นการลอกค่าที่เราให้ไปสร้างใหม่ จึงสามารถแยกเลิกหน่วยความจำที่จองไว้ได้ รวมทั้งหน่วยความจำที่เก็บภาพด้วย เช่นเมื่อโหลดจากไฟล์

นำออกแสดง

แผ่นภาพที่ได้นี้จะมีรูปแบบเดียวกับที่ใช้ LoadBitmap ในการโหลดรีเซอร์สของบิตแม็พ จึงใช้โปรแกรมเดียวกันได้ดังโปรแกรม DispBMPs ในลิสติง 1.4 (ขอให้ป้อนไว้หน้าลิสติง 1.2) ซึ่งได้กล่าวในรายละเอียดไว้แล้วในเรื่อง "ไอคอนของขวัญ" จึงขอกล่าวถึงการทำงานโดยสังเขป คือ

- ขอค่าดีซีของจอภาพ นำมาทำเป็นดีซีเทียม (ไม่มีดีไวซ์ไดรเวอร์)
- ให้แผ่นภาพกับดีซีเทียม
- ลอกแผ่นภาพจากดีซีเทียมไปให้แก่ดีซีของจอภาพ ด้วยวิธีย่อขยาย (StretchBlt) หรือลอกเท่าขนาดของเดิม (BitBlt) จะปรากฏภาพบนจอภาพ

ซึ่งใน wmRRultonDown ผมให้แสดง 2 ครั้ง ครั้งแรกเท่าขนาดของภาพ ครั้งที่สองขยาย 16 เท่า

โปรแกรมที่สมบูรณ์

ขอให้ป้อนโปรแกรมตามลิสติงดังกล่าวมา แล้วต่อท้ายด้วยโปรแกรมของสกุล TheAppC กับโปรแกรมหลักในลิสติง 1.5 จะได้โปรแกรมที่สมบูรณ์ครับ

ยูนิต Utem

เพื่อแก้ปัญหาดังกล่าวมาจะเขียนโปรแกรม โหลดไฟล์บิตแม็พลงหน่วยความจำและทำเป็นแผ่นภาพ (เพื่อให้เหมือนกับที่ใช้ฟังก์ชัน LoadBitmap โหลดรีซอร์สของบิตแม็พ) และเพื่อให้สมบูรณ์จะเขียนโปรแกรมสำหรับเซฟแผ่นภาพลงไฟล์บิตแม็พด้วย

ซึ่งความจริงแล้วในวินโดวส์น่าจะมีฟังก์ชันเพื่อดำเนินการในเรื่องนี้และความจริงอีกเหมือนกันในชุดของ

```
(* ------------------ TheAppC Class ---------------- *)

TYPE
 TheAppC = OBJECT(TApplication)
   procedure  InitMainWindow; virtual;
        END;


PROCEDURE  TheAppC.InitMainWindow;
BEGIN
 MainWindow := New(WndProCP, Init(nil,thetitle));
END;

(* ------------------ MAIN PROGRAM ----------------- *)

VAR  app :  TheAppC;

BEGIN
 app.Init(appname);
 app.Run;
 app.Done;
END.

attr.menu := LoadMenu(HInstance, menuname);


END.
```

ลิสติงที่ 1.5

```
UNIT  UTem;

Interface

USES  WinTypes,WinProcs,WObjects,Strings;

TYPE
IOFunction = function(FP: integer; Buf: PChar; Size: Integer): Word;
procedure AHIncr;

function   LoadBitmapFile(FileName: PChar;
                var width,height : Longint): HBitmap;
function   SaveBitmapFile(FileName: PChar; hBmp: HBitmap): Integer;
function   HugeIO(IOFunc: IOFunction; hFl: Integer;
             buf: Pointer; Size: Longint) : Word;
function   _LFileSize(hFl : integer) : longint;
function   bmAlignDouble(Size: Longint): Longint;


Implementation
```

ลิสติงที่ 2.1

เทอร์โบปาสคาลมีโปรแกรมย่อยเดียวกันนี้ ซึ่งสำหรับเทอร์โบปาสคาล 1.5 อยู่ในยูนิต Bitmaps (ไฟล์ C : \TPW \OWLDEMOS \PAINT \BITMAPS.PAS)

ซึ่งผมเห็นว่าสามารถนำวิธีการไปใช้กับไฟล์ชนิดอื่นได้ จึงขอนำมากล่าวถึงเพื่อการศึกษาครับ แต่เมื่อนำมาแล้วต้องขอเปลี่ยนชื่อตัวแปร เพื่อให้ (ผม) จำได้ง่าย และปรับปรุงโปรแกรมบ้าง

โดยจะเก็บไว้ในยูนิต Utem (ที่ให้ชื่อนี้เพราะจะมีโปรแกรมอื่นอีกตามที่จะกล่าวในฉบับต่อไป) มีส่วนอินเตอร์เฟส (interface) ตามลิสติง 2.1 และรายละเอียดตามที่จะกล่าวต่อไปคือ

โปรแกรมสนับสนุน

ขอนำโปรแกรมย่อยที่เขียนขึ้นเพื่อสนับสนุนโปรแกรมอื่นในยูนิตนี้มากล่าวถึงก่อน ตามลิสติง 2.2 คือ

- DispErr เพื่อการแสดงข้อความบ่งถึงเหตุขัดข้อง (error)

- bmAllignDouble เป็นการหาร 8 ในแบบปัดเศษขึ้น เพื่อให้ผลหารหารด้วย 4 ลงตัว

ฟังก์ชันลึกลับ

ในวินโดวส์มีฟังก์ชันลึกลับ (mystery) อยู่อย่างน้อย 1 ฟังก์ชัน คือ AHInc เป็นฟังก์ชันที่ไม่ส่งค่ากลับ คือเป็นโพรซีเดอร์ในภาษาปาสคาล ที่เรียกว่าลึกลับเพราะมีอยู่ในไลบรารีกลางของวินโดวส์ แต่ไม่มีในคู่มือ นอกจากนี้การใช้ฟังก์ชันนี้ยังพิศดารอีกด้วยตามที่จะกล่าวต่อไป

เมื่อจะนำมาใช้งานต้องดำเนินตามวิธีที่จะใช้ฟังก์ชันในไลบรารีกลางคือ DLL (Dynamic Link Libraries) ดัง

```
(* ---------- General Routine ---------- *)

procedure AHIncr; external 'KERNEL' index 114;


PROCEDURE  DispErr(err : Integer);
VAR  txt : PChar;
BEGIN
 case  err of
  2   : txt := 'File not found';
  3   : txt := 'Path not found';
  4   : txt := 'Too many open files';
  5   : txt := 'File access denied';
  6   : txt := 'Invalid file handle';
  1001 : txt := 'File not found.';
  1002 : txt := 'Not bitmaps file.';
  1003 : txt := 'Not enough memory.';
  1004 : txt := 'Vital error.';
 else
  txt := 'File error';
 end;
 MessageBox(0,txt,'ERROR',mb_IconStop + mb_OK);
END;


function bmAlignDouble(Size: Longint): Longint;
begin
  bmAlignDouble := (Size + 31) div 32 * 4;
end;
```

ลิสติงที่ 2.2

```
function _lread(FileHandle : Integer; Buffer : PChar;
          Bytes : Integer) : Word;

function _lwrite(FileHandle : Integer; Buffer : PChar;
          Bytes : Integer) : Word;
```

รูปที่ 2 ส่วนหัวของฟังก์ชันโหลดและเซฟข้อมูลกับไฟล์

เช่นที่ในเทอร์โบปาสคาลทำกับทุกฟังก์ชันของวินโดวส์ เว้นฟังก์ชันนี้ (ดูในไฟล์ C:\TPW\OWL\WINPROCS.PAS)

คือแสดงส่วนหัวไว้ในส่วนอินเตอร์เฟสเช่นเดียวกับโปรแกรมย่อยอื่น ๆ ในยูนิต สำหรับในส่วนอิมพลีเมนต์แทนที่จะเป็นโปรแกรม จะแสดงอินเด็กซ์ของฟังก์ชันนี้ในไลบรารีกลาง (อยู่ตอนต้นของลิสติง 2.2)

ในทำนองเดียวกันเราสามารถเขียนโปรแกรมด้วยภาษาปาสคาลทำเป็นไลบรารีกลางไว้ ให้ภาษาอื่น เช่น ซี รวมทั้งเทอร์โบปาสคาลด้วยกันเอง ด้วยการนำไปใช้ คือ กลับกันครับ

การโหลดและเซฟข้อมูลขนาดใหญ่

ในวินโดวส์มีฟังก์ชัน __lRead และ __lWrite เพื่อโหลดและเซฟข้อมูลตามลำดับ ดังแสดงส่วนหัวไว้ในรูปที่ 2 เทียบได้กับ BlockRead และ BlockWrite ในเทอร์โบปาสคาล ค่าที่ผ่านไปให้คือ

- ค่าแฮนเดิลของไฟล์
- พอยน์เตอร์ไปยังบัฟเฟอร์ที่จะรับหรือเก็บข้อมูล
- ขนาดของข้อมูลที่จะให้โหลดหรือเซฟเป็นไบต์

จะเห็นว่าได้ให้แบบข้อมูลของขนาดเป็น Integer นั่นคือจะอ่านหรือเขียนกับไฟล์ได้ครั้งละไม่เกิน $7FFF หรือ 32 กิโลไบต์ลบ 1 เมื่อต้องการโหลดหรือเซฟข้อมูลมากกว่านี้ จะต้องใช้ฟังก์ชันนี้หลายครั้ง

จึงได้ทำเป็นฟังก์ชัน HugeIO ในลิสติง 2.3 ไว้ทั้งเพื่อการอ่านและเขียนกับไฟล์ โดยให้ผ่านค่าเป็นฟังก์ชัน __lRead หรือ __lWrite มาให้ กับค่าที่จะให้แก่ฟังก์ชันนี้ ซึ่งต่อไปจะขอเรียกว่า "ฟังก์ชันอ่านเขียน"

โดยจะขออธิบายการทำงานของฟังก์ชันนี้เป็นส่วนรวมด้วยตัวเลขง่าย ๆ คือ สมมติว่าต้องการอ่านหรือเขียนข้อมูล 50 ไบต์กับไฟล์ และให้อ่านหรือเขียนครั้งละไม่เกิน 15 ไบต์กับบัฟเฟอร์ขนาด 50 ไบต์ จะดำเนินการได้ดังนี้คือ

- ค่าพอยน์เตอร์ของบัฟเฟอร์ที่จะผ่านไปให้แก่ฟังก์ชันอ่านเขียน ครั้งแรกจะเป็นต้นบัฟเฟอร์คือช่องที่ 0 ครั้งต่อ ๆ ไปจะเป็นของช่องที่ 15, 30 และ 45 ตามลำดับ
- ค่าใน lft ซึ่งใช้จดจำตำแหน่งก็จะมีค่าเป็น 0, 15, 30 และ 45 ตาม

```
(* ------------------- Load and Store BMPs -------------------*)

CONST
 MaxIO  = 65534;
 OneIO  = 32768;
 BMType = $4D42;  { 'BM' }

TYPE
 PtrRec = record  Lo, Hi: Word  end;


FUNCTION HugeIO(IOFunc: IOFunction; hFi: Integer;
                buf: Pointer; Size: LongInt) : Word;
VAR
 lft, rde: LongInt;
BEGIN
 HugeIO := 1;
 lft := 0;
 while lft < Size do
 begin
  rde := Size - lft;
  if rde > OneIO then rde := OneIO;
  if IOFunc(hFi,
    Ptr(PtrRec(buf).Hi + PtrRec(lft).Hi * Ofs(AHIncr),PtrRec(lft).Lo),
    Integer(rde))
    <> rde then
  begin  HugeIO := 0;  Exit  end;
  Inc(lft, rde);
 end;
END;
```

ลิสติงที่ 2.3

ลำดับเช่นกัน

- ค่าใน rde คือจำนวนไบต์ที่ให้อ่านจะมีค่าเป็น 15, 15, 15 และ 5 ตามลำดับ

โดยหลักการมีเท่านี้ครับ แต่เมื่อต้องดำเนินการกับบัฟเฟอร์ที่ข้ามเซกเมนต์ จึงมีวิธีเฉพาะต้องขอให้ดูในรายละเอียดครับ คือ

- Ptr เป็นฟังก์ชัน Ptr (เซกเมนต์, ออฟเซต) แปลงค่าที่ให้เป็นเซกเมนต์กับออฟเซตเป็นพอยน์เตอร์

- Seg (ไม่ได้ใช้ในโปรแกรมนี้) กับ Ofs เป็นฟังก์ชันที่ให้ค่าเซกเมนต์และออฟเซตของสิ่งใดก็ได้ ที่มีตำแหน่งในหน่วยความจำ เช่น ตัวแปร หรือโปรแกรมย่อย ดังเช่น Ofs (AHIncr) จะให้ค่าออฟเซตของฟังก์ชันนี้ในหน่วยความจำ

- PtrRec เป็นแบบข้อมูลที่จะแยกค่าในตัวแปรขนาด 4 ไบต์ คือ LongInt และ Pointer ออกเป็นเวิร์ดสูงกับเวิร์ดต่ำคือ Hi กับ Lo ตามลำดับ การใช้จะใช้ในแบบ type cart เช่น PtrRec (buf) .Hi และโดยที่ buf เป็นพอยน์เตอร์ ค่าที่ได้คือค่าเซกเมนต์ของบัฟเฟอร์คือหน่วยความจำที่ buf ชี้อยู่ตามที่จองไว้ คือจะตรงกับ Seg (buf^) แต่ไม่ตรงกับ Seg (buf)

บรรทัดที่ขึ้นต้นด้วยคำว่า Ptr ทั้งหมดจะแปลได้ความว่า

- ให้นำค่าเซกเมนต์ของบัฟเฟอร์บวกด้วยค่าเวิร์ดสูงของ lft ซึ่งคูณด้วยค่าออฟเซตของฟังก์ชัน AHIncr มาเป็นค่าเซกเมนต์

- และนำเวิร์ดต่ำของ lft มาเป็นค่าออฟเซต

- ให้ Ptr ทำเป็นพอยน์เตอร์

จะได้พอยน์เตอร์ไปยังหน่วยความจำเมื่อต้องข้ามเซกเมนต์ตามต้องการ (เป็นปัญหาเรื้อรังของดอสกับ 8088) ที่ผมว่าฟังก์ชัน AHIncr เป็นฟังก์ชันลึกลับ เพราะฟังก์ชันนี้ทำหน้าที่อะไรนอกจากคนสร้างมันไม่มีใครทราบ การนำมาใช้ก็พิศดาร คือไม่ได้เรียกใช้ฟังก์ชันนี้ กลับนำแอดเดรสมาใช้

อีกประเด็นหนึ่งที่บางท่านอาจสงสัยว่า ทำไมไม่นำค่าเวิร์ดต่ำใน buf คือค่าออฟเซตของบัฟเฟอร์มาใช้ เดาได้อย่างเดียวครับคือ GlobalAlloc จองหน่วยความจำโดยให้ค่าออฟเซตเป็นศูนย์ ซึ่งตรงกับการทดลองตรวจสอบ

เก็บแผ่นภาพลงไฟล์

การเก็บบิตแม็พลงไฟล์สามารถเก็บได้ทั้งที่เป็นภาพและเป็นแผ่นภาพ แต่ผมต้องการให้เก็บจากแผ่นภาพเพื่อใช้ในงานอื่น ที่นำมาแสดงก็เพื่อให้เห็นการสร้าง (ในโปรแกรม) การเก็บ และการนำมาใช้ครับ

ก็ต้องขอเรียนให้ทราบก่อนว่า ถ้าหากเราสร้างภาพและทำเป็นแผ่นภาพดังกล่าวมา แล้วนำแผ่นภาพมาเก็บจะมีขั้นตอนที่ซ้ำซ้อนกัน ทั้งนี้เพราะเมื่อเราให้ค่าเรคอร์ดเล็กแก่ CreateDIBitmap ฟังก์ชันนี้จะเก็บค่าบางส่วนของเรคอร์ดเล็กเท่านั้นไว้กับแผ่นภาพ เราต้องให้ค่าเรคอร์ดนี้ใหม่

นั่นคือ หากต้องการเก็บภาพลงไฟล์ก็เพียงแต่นำค่าที่เตรียมไว้ให้แก่

CreateDIBitmap ไปเก็บลงไฟล์ และเช่นกันโปรแกรมเก็บแผ่นภาพลงไฟล์ก็จะเขียนทำนองเดียวกับโปรแกรมทำภาพให้เป็นแผ่นภาพ

โดยที่ถ้าต้องการให้เป็นไฟล์ .BMP ตามมาตรฐานของวินโดวส์ จะต้องเพิ่มส่วนหัวของไฟล์ดังโปรแกรม SaveBitmapFile ในลิสติง 2.4 เป็นฟังก์ชันให้ค่าเป็น 0 เมื่อมีข้อผิดพลาด และเป็น 1 เมื่อถูกต้อง โดยให้ผ่านไฟล์ที่จะใช้เก็บกับแผ่นภาพมาให้ เพื่อดำเนินการดังนี้คือ

1. อ่านค่าที่กล่าวไว้ว่าเป็นบางส่วนของเรคอร์ดเล็กจากแผ่นภาพ (hBmp) ให้แก่ abmp ซึ่งเป็นตัวแปรแบบ TBitmap เพื่อจะนำค่าต่าง ๆ มาใช้

2. หาค่า bitcnt (ขอให้อ่านว่า bit count) คือจำนวนบิตต่อ 1 จุดภาพ ซึ่งค่า 2 ยกกำลัง bitcnt (1 shl bitcnt) คือจำนวนช่องของจานสี เมื่อนำไปคูณกับขนาดของช่องจะได้ขนาดของจานสี ให้ค่านี้แก่ cosi แต่ถ้า bitcnt เป็น 24 จะไม่มีจานสี เพราะจุดภาพหนึ่ง ๆ ใช้หน่วยความจำ 3 ไบต์ เป็นค่า RGB ได้พอดี

3. หาขนาดของภาพเป็นไบต์ ด้วยการนำความกว้างของภาพเป็นจุดภาพคูณด้วย bitcnt แล้วหารด้วย 8 แต่เป็นการหารแบบปัดเศษขึ้นดังกล่าวมา จะได้ความกว้างของภาพเป็นไบต์คูณด้วยความสูง จะได้ขนาดของหน่วยความจำทั้งหมดที่จะเก็บภาพนี้ ให้ค่าแก่ pctsi

4. เปิดไฟล์ในแบบสร้างไฟล์ใหม่ จะได้ค่าแฮนเดิลของไฟล์ใน hFi

5. จองหน่วยความจำเท่ากับขนาดของเรคอร์ดเล็กบวกด้วยขนาดของจานสี แต่ให้ค่าหน่วยความจำนี้แก่ pInfo ซึ่งเป็นเรคอร์ดใหญ่

ลิสติงที่ 2.4

```
FUNCTION SaveBitmapFile(FileName: PChar; hBmp: HBitmap): Integer;
VAR abmp : TBitmap; fihead : TBitmapFileHeader;
    pInfo : PBitmapInfo; adc : HDC;
    hInfo : THandle; buf : Pointer;
    cosi, pctsi : LongInt; bitcnt : Word;
    hFi : Integer;
BEGIN
  SaveBitmapFile := 0;
  if GetObject(hBmp, SizeOf(TBitmap), @abmp) = 0 then Exit;

  bitcnt := abmp.bmPlanes * abmp.bmBitsPixel;
  if (bitcnt <> 24) then
    cosi := SizeOf(TRGBQuad) * (1 shl bitcnt)
  else
    cosi := 0;
  pctsi := bmAlignDouble(abmp.bmWidth * bitcnt) * abmp.bmHeight;

  hFi := _lcreat(FileName, 0);
  if hFi = -1 then Exit;

  GetMem(pInfo, SizeOf(TBitmapInfoHeader) + cosi);
  if pInfo <> nil then
    begin
    with pInfo^.bmiHeader do
      begin
      biSize := SizeOf(TBitmapInfoHeader);
      biWidth := abmp.bmWidth; biHeight := abmp.bmHeight;
      biPlanes := 1;          biBitCount := bitcnt;
      biCompression := 0;     biSizeImage := pctsi;
      biXPelsPerMeter := 0;   biYPelsPerMeter := 0;
      biClrUsed := 0;         biClrImportant := 0;
      end;

    with fihead do
      begin
      bfOffBits := SizeOf(fihead) + SizeOf(TBitmapInfo) + cosi;
      bfType := BMType;  bfSize := bfOffBits + pctsi;
      bfReserved1 := 0; bfReserved2 := 0;
      end;

    hInfo := GlobalAlloc(gmem_Fixed, pctsi);
    if hInfo <> 0 then
      begin
      buf := GlobalLock(hInfo);

      adc := GetDC(0);
      if GetDIBits(adc, hBmp, 0, pctsi, buf, pInfo^, dib_RGB_Colors) <> 0 then
        begin
        ReleaseDC(0, adc);
        { Write to file }
        _lwrite(hFi, @fihead, SizeOf(fihead));
        _lwrite(hFi, PChar(pInfo), SizeOf(TBitmapInfo) + cosi);
        HugeIO(_lwrite, hFi, buf, pctsi);
        SaveBitmapFile := 1;
        end;
```

ลิสติงที่ 2.4 (ต่อ)

```
        GlobalUnlock(hInfo);
        GlobalFree(hInfo);
        end;

      FreeMem(pInfo, SizeOf(TBitmapInfoHeader) + cosi);
     end;

   _lclose(hFi);
  END;
```

6. ให้ค่าต่าง ๆ แก่เรคอร์ดเล็กตามที่เคยกล่าวมาแล้ว

7. เตรียมส่วนหัวของไฟล์ให้แก่ fihead คือ

- ให้ตำแหน่งของภาพจากต้นไฟล์แก่ฟิลด์ ofOffBits
- ให้อักขระ BM แก่ฟิลด์ bfType เพื่อการตรวจสอบ
- ให้ขนาดของไฟล์ทั้งหมดแก่ฟิลด์ bfSize

8. จองหน่วยความจำขนาดเท่าภาพด้วย GlobalAlloc ซึ่งใช้สำหรับจองหน่วยความจำขนาดใหญ่ที่เป็นส่วนกลางของวินโดวส์ ได้ค่าแฮนเดิลแก่ hInfo แล้วทำเป็นพอยน์เตอร์ให้แก่ buf

9. ฟังก์ชัน GetDIBits ดำเนินการตรงข้ามกับ CreateDIBimaps คือจะให้ภาพในแผ่นภาพแก่ buf และจะให้จำนวนสีแก่จานสีที่ติดอยู่กับ pInfo ซึ่งเป็นเรคอร์ดใหญ่ ส่วนค่าในเรคอร์ดเล็กเราต้องให้ใหม่ และต้องให้ก่อนด้วย เพราะ GetDIBits จะนำค่าในฟิลด์ biSize ไปหาตำแหน่งเริ่มต้นของจานสีไป pInfo

10. จากนี้จะเป็นการเก็บลงไฟล์ คือ

- เก็บส่วนหัวของไฟล์
- เก็บเรคอร์ดใหญ่รวมทั้งจานสีทั้งหมดด้วย
- เก็บตัวภาพด้วยการเก็บข้อมูลขนาดใหญ่

อ่านแผ่นภาพจากไฟล์

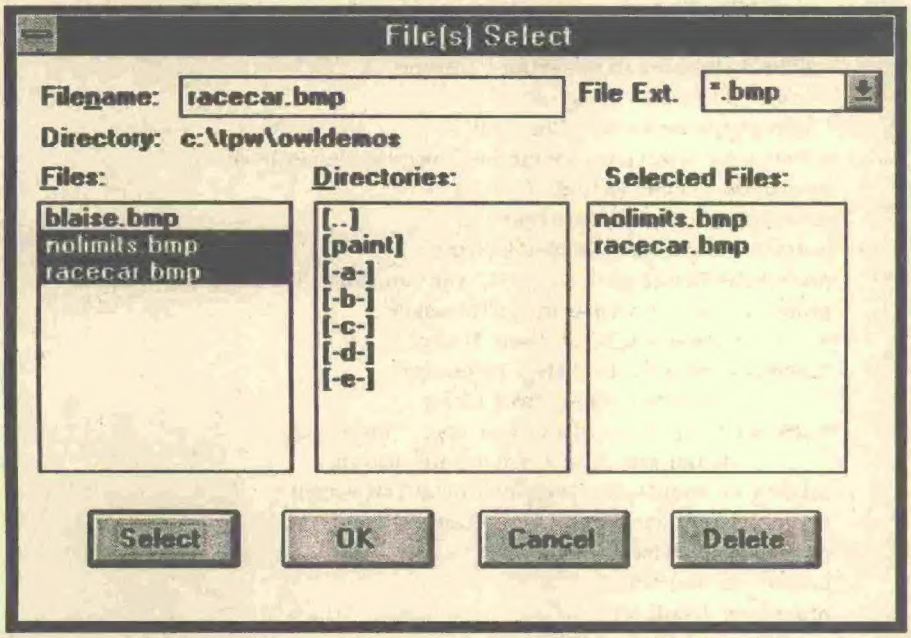
สำหรับการอ่านภาพจากไฟล์ .BMP มาทำเป็นแผ่นภาพตามฟังก์ชัน LoadBitMapFile ในลิสติง 2.5 ดำเนินการตรงข้ามกับการเก็บผสมกับการทำภาพเป็นแผ่นภาพ จึงของดคำอธิบายครับ

ลิสติงที่ 2.5

```
FUNCTION _LFileSize(hFi : integer) : longint;
VAR CurPos : longint;
BEGIN
 CurPos := _llseek(hFi,0,1);
 _LFileSize := _llseek(hFi,0,2);
 _llseek(hFi,CurPos,0);
END;



FUNCTION LoadBitmapFile(FileName: PChar; var width,height : LongInt): HBitmap;
VAR hFi: Integer;  hMem: THandle;  adc: HDC;
    datsi, pctofs: Longint;
    pInfo: PBitmapInfo;  fihead: TBitmapFileHeader;

BEGIN
 LoadBitmapFile := 0;
 hFi := _LOpen(FileName, of_Read);
 if hFi = -1 then  begin  DispErr(1001);  Exit  end;

 if (_LRead(hFi, @fihead, SizeOf(fihead)) <> SizeOf(fihead)) or
   (fihead.bfType <> BMType) then
  begin  _LClose(hFi);  DispErr(1002);  Exit  end;

 datsi := _LFileSize(hFi) - SizeOf(fihead);
 hMem := GlobalAlloc(gmem_Moveable, datsi);
 if hMem = 0 then
  begin  _LClose(hFi);  DispErr(1003);  Exit  end;
 pInfo := GlobalLock(hMem);

 if (HugeIO(_LRead, hFi, pInfo, datsi) <> 0) and
   (pInfo^.bmiHeader.biSize = SizeOf(TBitmapInfoHeader)) then
  begin
  pctofs := fihead.bfOffBits - SizeOf(fihead);
  adc := GetDC(0);
  LoadBitmapFile := CreateDIBitmap(adc, pInfo^.bmiHeader,
    cbm_Init, Ptr(PtrRec(pInfo).Hi,pctofs),pInfo^, dib_RGB_Colors);
  width  := pInfo^.bmiHeader.biWidth;
  height := pInfo^.bmiHeader.biHeight;
```

ลิสติงที่ 2.5 (ต่อ)

```
    ReleaseDC(0, adc);
    end
   else DispErr(1004);

  GlobalUnlock(hMem);
  GlobalFree(hMem);
  _LClose(hFl);
END;
```



รูปที่ 3 การใช้มัลติไฟล์ไดอะล็อกบ็อกซ์ เช่นเดียวกับไฟล์ไดอะล็อกบ็อกซ์ทั่วไปเช่นของเทอร์โบปาสคาล แตกต่างกันตรงสามารถเลือกไฟล์ได้หลายไฟล์ โดย

- เลือกไฟล์ในกรอบ Files ตามต้องการ แล้วเลือกปุ่ม Select จะปรากฏชื่อไฟล์ที่เลือกในกรอบ Selected Files จะเป็นการเลือกไฟล์ตามลำดับอักษร หากต้องการให้เป็นไปตามลำดับที่เลือกให้เลือกไฟล์ และปุ่ม Select ครั้งละ 1 ไฟล์

- หากต้องการเปลี่ยนไดเรคทอรีเพื่อเลือกไฟล์อื่นต่อไปอีก จะต้องให้ชื่อไฟล์ในกรอบ Filname มีไวล์การ์ดโดยป้อนหรือเลือกจากกรอบ File Ext. ก่อน (เช่นตามรูปขณะนี้จะเปลี่ยนไดเรคทอรีไม่ได้)

ทำเป็นยูนิต

ขอให้ป้อนโปรแกรมตามลิสติง 2.1 ถึง 2.5 ลงไฟล์ UTEM.PAS แล้วคอมพายล์เป็นไฟล์ UTEM.TPU เพื่อจะนำไปใช้งานต่อไปครับ

ยูนิต UFiDlg

เรื่องราวของยูนิตนี้ผมกล่าวไว้ในไมโครฯ ฉบับที่ 91 และจะต้องนำมาใช้กับโปรแกรม SlideShow ตามที่จะกล่าวต่อไปด้วย โดยจะนำ EDitNumDlgC กับ MultiFiDlgC มาใช้ในบทความตอนนี้

โปรแกรม SlideShow

จุดประสงค์ของการเขียนโปรแกรมนี้ เพื่อแสดงการเขียนโปรแกรมบนวินโดวส์ดังกล่าวและที่จะกล่าวต่อไป เช่น การใช้พื้นที่ใช้งาน (Client area) เต็มจอภาพ แต่ก็เป็นโปรแกรมที่นำไปใช้งานได้จริงโปรแกรมหนึ่ง ซึ่งจะแสดงภาพบิตแม็พเต็มจอภาพผลัดเปลี่ยนหมุนเวียนไปตามภาพที่เลือกไว้

การแสดงภาพบิตแม็พเต็มจอภาพทำให้ไม่มีกรอบหรือขอบใด ๆ ของหน้าต่างแสดงเมนูไม่ได้ (นอกจากจะใช้เมนูลอย) การเลือกการทำงานของโปรแกรมจึงต้องใช้คีย์บอร์ดกับเม้าส์เท่านั้น โดยที่ผมได้กำหนดไว้ว่า

คีย์บอร์ด

- กดคีย์ enter ให้แสดงมัลติไฟล์ไดอะล็อกบ็อกซ์ เพื่อให้ผู้ใช้เลือกไฟล์ 1 หรือหลายไฟล์ที่ต้องการนำออกแสดง (การใช้งานไดอะล็อกบ็อกซ์นี้อยู่ในรูปที่ 3)

- กด space bar จะแสดงไดอะล็อกบ็อกซ์ให้ผู้ใช้ป้อนห้วงเวลาของการแสดงภาพแต่ละภาพเป็นส่วนสิบของวินาที เช่น 15 คือ 1.5 วินาที (หากไม่เลือกจะใช้ค่าที่ตั้งไว้คือ 1 วินาที)

- กดคีย์ esc ยุติการทำงานของโปรแกรมนี้

เม้าส์

- คลิ๊กเม้าส์ซ้ายหรือขวา หมายถึงให้หยุดภาพ ต่อเมื่อคลิ๊กเม้าส์ขวาหรือซ้าย (คือกลับกัน) ให้เดินภาพต่อไป

- หากต้องการเลื่อนภาพถอยหลังให้คลิ๊กเม้าส์ซ้ายหยุด แล้วคลิ๊กเม้าส์ซ้ายเพื่อเลื่อนภาพถอยหลังครั้งละ 1

ภาพ คลิ๊กเม้าส์ขวาเพื่อเดินภาพต่อไป

- หากต้องการเลื่อนภาพเดินหน้าให้คลิ๊กขวาแทนและกลับกัน

นำไปเขียนโปรแกรมดังการกำหนดสกุลของหน้าต่างหลักในลิสติง 3.1 มีรายละเอียดของโปรแกรมดังนี้ คือ

เต็มจอภาพ

การขยายหน้าต่างให้มีพื้นที่ใช้งานเต็มจอภาพทำใน Init ในลิสติง 3.2 คือที่ให้ค่า x และ y ของ attr เป็น (0, 0) และให้มีความกว้างและความสูงเท่ากับความกว้างและความสูงของจอภาพ และให้ค่า style เป็น ws__Popup เท่านั้นไม่มีรายละเอียดอื่นอีก

สำหรับค่า ws__Maximize ที่ทำเป็นหมายเหตุไว้ เพราะควรจะ or ด้วยค่านี้ด้วย แต่เมื่อไม่ให้ค่านี้ก็ได้หน้าต่างตามต้องการ จึงทำเป็นหมายเหตุไว้เพื่อทราบ

หลังจากรันโปรแกรมนี้แล้วจอภาพจะเป็นพื้นขาวเต็มจอภาพรอการใช้งานของผู้ใช้ จึงให้เมสเสจบ็อกซ์แสดงการใช้คีย์ใน SetUpWindows (ซึ่งจะแสดงไดอะล็อกบ็อกซ์ก่อนที่จอภาพจะเป็นสีขาว)

เมื่อผู้ใช้กดคีย์

หลังจากรันโปรแกรมนี้แล้วจอภาพจะเป็นพื้นขาวเต็มจอภาพ เมื่อผู้ใช้กดคีย์จะมาทำงานที่ wmChar ในลิสติง 3.3 นำค่าคีย์มาตรวจคือ

- ถ้าเป็น space bar ให้ผู้ใช้ป้อนห้วงเวลาใหม่ด้วย NewTime
- ถ้าเป็น enter ให้ผู้ใช้เลือกหรือถอนไฟล์ด้วย GetUserSelectedFile
- ถ้าเป็น esc ให้ปิดหน้าต่างยุติการทำงานของโปรแกรมนี้

```
PROGRAM  SlideShow;
USES  WinTypes,WinProcs,WObjects,Strings,ufidlg,utem;
CONST  appname = 'slideshw';
      thetitle = 'SLIDE SHOW';

 timerID = 1;   tmemul = 100;


TYPE
 WndProcCP = ^WndProcC;
 WndProcC = OBJECT(TWindow)
  curBMP  : HBitmap;
  spoon : PStrCollection;
  scrwid,scrhi,nextfi,pausetme : Integer;
  bmpwid,bmphi  : Longint;
  lpause,rpause,ready : Boolean;
 constructor  Init(apar : PWindowsObject; atitle : PChar);
 destructor   Done; virtual;
 procedure  SetupWindow;  virtual;
 function   CanClose : Boolean; virtual;
 procedure  Paint(paintDC : HDC; var paintInfo : TPaintStruct); virtual;
 procedure  wmTimer(var msg : TMessage);
          virtual wm_First + wm_Timer;
 procedure  wmChar(var Msg: TMessage);
          virtual wm_First + wm_Char;
 procedure  wmLButtonDown(var msg : TMessage);
          virtual wm_First + wm_LButtonDown;
 procedure  wmRButtonDown(var msg: TMessage);
         virtual wm_First + wm_RButtonDown;
 procedure  GetUserSelectedFile;
 procedure  NewTime;
 procedure  LoadBMP;
 procedure  ShowForward;
 procedure  ShowBackWard;
        END;
```

ลิสติงที่ 3.1

ลิสติงที่ 3.2

```
(* ------------------  WndProcC method  ---------------- *)

CONSTRUCTOR  WndProcC.Init(apar : PWindowsObject; atitle : PChar);
BEGIN
 TWindow.Init(apar,atitle);
 scrwid := GetSystemMetrics(sm_CXScreen);
 scrhi  := GetSystemMetrics(sm_CYScreen);
 attr.x := 0;  attr.y := 0;  attr.w := scrwid;  attr.h := scrhi;
 attr.style := ws_PopUp ( or ws_Maximize) ;
 curBMP := 0;  nextfi := 0;  pausetme := 15;
 lpause := false;  rpause := false;  ready := False;
 spoon := New(PStrCollection, Init(10, 5));
END;

DESTRUCTOR  WndProcC.Done;
VAR   txt : array[0..19] of Char;
BEGIN
```

ลิสติงที่ 3.2 (ต่อ)

```
 Dispose(spoon, Done);
 TWindow.Done;
END;


PROCEDURE WndProC.SetupWindow;
BEGIN
 TWindow.SetupWindow;
 MessageBox(HWindow,'enter = Select files.  space = Set time.'
            + #10#13 + 'esc = Exit.', 'Slide Show', mb_OK);
 ShowCursor(False);
END;


FUNCTION WndProC.CanClose : Boolean;
BEGIN
 KillTimer(HWindow, timerID);
 ShowCursor(True);
 CanClose := TWindow.CanClose;
END;
```

```
PROCEDURE WndProC.wmTimer(var msg : TMessage);
BEGIN
 if lpause or rpause then Exit;
 ShowForward;
END;


PROCEDURE WndProC.wmChar(var Msg: TMessage);
BEGIN
 case msg.wParam of
  vk_Space  :  NewTime;
  vk_Return :  GetUserSelectedFile;
  vk_Escape :  CloseWindow;
 else
   DefWndProc(msg);
 end; { case}
END;


PROCEDURE WndProC.NewTime;
VAR numed : EditNumDlgCP;  tem : Integer;
BEGIN
 if Application^.ExecDialog(New(EditNumDlgCP,
    Init(@self, 'Enter time in tenth of second', 'TIME INTERVAL',
          '1', 0, pausetme))) <> id_Ok then Exit;
 if spoon^.count <> 0 then
   begin
   KillTimer(HWindow, timerID);
   SetTimer(HWindow, timerID, tmemul * pausetme, nil);
   end;
END;
```

ลิสติงที่ 3.3

- ถ้าเป็นคีย์อื่น ๆ ส่งไปให้วินโดส์ดำเนินการ

การตั้งเวลา

การตั้งเวลาดำเนินการใน NewTime โดยใช้ EditNumDlgC ในยูนิต UFiDlg แสดงไดอะล็อกบ็อกซ์ให้ผู้ใช้ป้อนเวลาให้แก่ตัวแปร pausetme และตั้งเวลาใหม่หากผู้ใช้เลือกไฟล์แล้ว

การเลือกไฟล์

เมื่อผู้ใช้กดคีย์ enter เพื่อเลือกไฟล์จะมาทำงานใน GetUserSelectFile ในลิสติง 3.4 ซึ่งจะให้ MultiFiDlgC อยู่ในยูนิต UFiDlg แสดงไดอะล็อกบ็อกซ์ให้ผู้ใช้เลือกไฟล์ โดยส่งที่เก็บข้อมูลคือ spoon สกุล TStrCollection ไปให้ จะได้ชื่อไฟล์รวมทั้งไดเรคทอรีกลับมา และจะเป็นผู้เริ่มเดินเวลา

โดยที่ผู้ใช้อาจเลือกรวมทั้งถอนไฟล์ได้อีก ตอนต้นของฟังก์ชันนี้จึงให้ตรวจว่าที่เก็บข้อมูลว่างหรือไม่ ถ้าไม่ว่างแสดงว่าเป็นการเลือกไฟล์ครั้งต่อ ๆ ไปจึงให้หยุดเวลา และเดินนาฬิกาใหม่หลังจากที่ผู้ใช้เลือกไฟล์แล้ว แล้วให้เริ่มต้นแสดงภาพแรกด้วย LoadBMP

การแสดงภาพ

เมื่อโปรแกรมใดต้องการให้แสดงภาพเช่น GetUserSelectFile จะเรียกใช้ LoadBMP ในลิสติง 3.5 ซึ่งมีการดำเนินการดังนี้

- ตรวจว่าในแผ่นภาพมีภาพหรือไม่ ถ้ามีให้ยกเลิก
- ตรวจว่าในที่เก็บข้อมูลมีข้อมูลหรือไม่ ถ้าไม่มีจะไม่ดำเนินการใดต่อ เพราะผู้ใช้อาจถอนไฟล์จนหมด
- อ่านไฟล์ตรงตำแหน่งที่ nextfi ชี้อยู่ (ถ้าเป็นครั้งแรกคือไฟล์แรกในที่

เก็บ นำไปให้ LoadBitmapFile โหลดแผ่นภาพให้)

- ถ้ามีข้อผิดพลาดจะปิดหน้าต่างยุติการรันของโปรแกรมนี้ (การแจ้งข้อผิดพลาด LoadBitmapFile เป็นผู้แจ้ง)

- หากถูกต้องให้ใช้ InvalidatiRect เพื่อลบภาพเดิม เป็นการบังคับให้ไปทำงานที่ Paint

การแสดงภาพจริง ๆ จึงอยู่ที่ Paint (ในลิสติง 3.5) ซึ่งจะ

- ตรวจว่ามีภาพให้แสดงหรือไม่ ถ้าไม่มีจะไม่ดำเนินการใด เพราะจะมีการเรียกใช้ Paint เมื่อเริ่มรัน

- คำนวณตัวคูณ n เพื่อให้ความกว้างหรือความสูงเต็มจอภาพ เพื่อรักษาสัดส่วนของภาพไว้ สำหรับบิตแม็พที่ไม่เต็มจอภาพซึ่งต้องการการขยาย

- นำออกแสดงโดยตรงหรือขยายตามวิธีที่เคยกล่าวมาแล้ว

การเปลี่ยนภาพ

เมื่อมีไทเมอร์อินเตอร์รัพต์เข้ามาจะทำงานที่ wmTimer (ในลิสติง 3.3) ซึ่งจะตรวจว่าเป็นช่วงหยุดหรือไม่ ถ้าไม่ใช่จะให้ ShowForward แสดงภาพต่อไป

เมื่อผู้ใช้คลิ๊กเม้าส์

เมื่อผู้ใช้คลิ๊กเม้าส์จะมีการทำงานในกิจกรรมต่าง ๆ ในลิสติง 3.6 ซึ่งจะเป็นไปตามที่ออกแบบไว้ครับ ที่เพิ่มมาคือให้ตรวจว่า ready หรือไม่ (ให้เป็นเท็จไว้ใน Init และให้เป็นจริงไว้ใน GetUserSelectFire) เพื่อยกเลิกการคลิ๊กเม้าส์ของผู้ใช้เมื่อยังไม่ได้แสดงภาพ

โปรแกรมทั้งหมด

ขอให้ป้อนโปรแกรมตามลิสติง

```
PROCEDURE WndProC.GetUserSelectedFile;
VAR   finme : array[0..79] of Char;
      extaray : PChar;
BEGIN
 if spoon^.count <> 0 then KillTimer(HWindow, timerID);
 ShowCursor(True); extaray := '*.bmp';
 if Application^.ExecDialog(New(MultiFlDlgCP,
    Init(nil,StrCopy(finme, 'c:\tpw\owldemos\*.bmp'),
     extaray, spoon))) = id_Ok then
  begin
  SetTimer(HWindow, timerID, tmemul * pausetme, nil);
  LoadBMP; ready := True;
  end;
 ShowCursor(False);
END;
```

ลิสติงที่ 3.4

```
PROCEDURE WndProC.LoadBMP;
VAR  finme : array[0..79] of Char;
BEGIN
 if curBMP <> 0 then DeleteObject(curBMP);
 if spoon^.count = 0 then Exit;
 StrCopy(finme,spoon^.At(nextfl));
 curBMP := LoadBitmapFile(finme, bmpwid, bmphi);
 if curBMP = 0 then
  begin CloseWindow; Exit end;
 InvalidateRect(HWindow,nil,true);
END;



PROCEDURE WndProC.Paint(paintDC : HDC;
                var paintInfo : TPaintStruct);
VAR  memDC : HDC; hOldObj : THandle; n,nw,nh : Integer;
BEGIN
 if spoon^.count = 0 then Exit;
 nw := scrwid div bmpwid; if nw = 0 then nw := 1;
 nh := scrhi div bmphi; if nw = 0 then nw := 1;
 if nw = nh then n := nw
 else if nw > nh then n := nh
 else n := nw;
 memDC := CreateCompatibleDC(paintDC);
 hOldObj := SelectObject(memDC, curBMP);
 if n = 1 then
  BitBlt(paintDC, 0,0,bmpwid,bmphi, memDC,0,0, srcCopy)
 else StretchBlt(paintDC, 0,0,bmpwid*n,bmphi*n,
          memDC,0,0,bmpwid,bmphi,srcCopy);
 SelectObject(memDC, hOldObj);
 DeleteDC(memDC);
END;
```

ลิสติงที่ 3.5

```
PROCEDURE WndProC.wmLButtonDown(var msg : TMessage);
BEGIN
 if not ready then Exit;
 if rpause then
  begin rpause := false; ShowForward end
 else if lpause then ShowBackWard
 else lpause := true;
END;


PROCEDURE WndProC.wmRButtonDown(var msg: TMessage);
BEGIN
 if not ready then Exit;
 if lpause then
  begin lpause := false; ShowForward end
 else if rpause then ShowForward
 else rpause := true;
END;


PROCEDURE WndProC.ShowForward;
BEGIN
 if nextfi = spoon^.count - 1 then nextfi := 0
 else inc(nextfi);
 LoadBMP;
END;


PROCEDURE WndProC.ShowBackward;
BEGIN
 if nextfi = 0 then nextfi := spoon^.count - 1
 else dec(nextfi);
 LoadBMP;
END;
```

ลิสติงที่ 3.6

3.1 ถึง 3.6 แล้วนำโปรแกรมในลิสติง 1.5 อันเป็นส่วนของสกุล TheAppC กับเมนโปรแกรมมาต่อท้าย (เหมือนกันทุกโปรแกรม)

ปัญหาของโปรแกรมคงไม่มี ปัญหาอยู่ที่จะนำภาพบิตแม็พที่ไหนมาแสดง ในการใช้งานจริงจะต้องเตรียมไว้ครับ เช่น วาดด้วย PaintBrush หรือสแกนเข้ามา

สำหรับในการทดลองในเทอร์โบ-ปาสคาลมีอยู่ 3 ภาพตามไดเรคทอรีที่ผมให้ไว้ในลิสติง 3.4 ของวินโดวส์โดยเฉพาะวินโดวส์ 3.1 จะมีอีกสิบกว่าภาพในไดเรคทอรี C:\WINDOWS ขอให้ทดลองดูครับถ้าเป็นภาพที่ไม่ต้องขยาย เช่น nolimits.bmp จะแสดงได้เร็วกว่า

ท้ายตอน

คิดว่าโปรแกรมนี้คงมีประโยชน์ตามสมควร โดยเฉพาะเพื่อเป็นจุดเริ่มต้นในการปรับปรุงต่อไป หากเป็นงานเฉพาะที่มีภาพตายตัวอาจไม่ต้องใช้ที่เก็บข้อมูลให้ทำเป็นอะเรย์ของไฟล์ ตั้งเวลาและให้แสดงภาพแรกใน SetupWindow เช่นกันอาจทำเป็นอะเรย์ของเวลาของแต่ละภาพ

พบกันใหม่ฉบับหน้าในเรื่องของเครื่องพิมพ์ซึ่งคล้าย ๆ จะลึกลับเช่นกัน (หากต้องการโปรแกรมของฉบับนี้สั่งซื้อได้จากบริการซอฟต์เมล์ครับ)